Data Wrangling

To study the relationship between money and results for European soccer teams, I leveraged two datasets. The first is a SQLite database containing individual match statistics for every game played in all of the top European leagues between 2008 and 2015. The second is a collection of CSV files containing individual details of player transfers, both purchases and sales, made by each team in all of the top European leagues between 1992 and 2019. When reading in the transfer data, I decided to keep the year outside of the match data on either end, so I looked at transfers made between 2007 and 2016.

The match data was held in a SQLite database. I used the sqlite3 and pandas package to read the necessary tables in as pandas dataframes. The table containing match data used only ID numbers for teams and leagues, so I needed to also read in the tables containing the corresponding team and league names. I created dictionaries consisting of team ID : team name as well as league ID : league name, as key, value pairs in order to add the correct team and league name to each row in the match dataset. I also added three boolean columns, home_win, away_win, and draw. One of the three columns in each row would read True depending on the result of the match. This would be used to total a given team's record at the end of a season.

The only column in this dataset which contained an inconsistent data type was the year column. In soccer, seasons span across two years and are typically referenced by both years. For example, the 2008/2009 season references the season that starts in late 2008 and spans until mid-2009. Because of this, this column consisted of strings containing both years ("2008/2009", "2009/2010", etc.). To fix this, I changed the year to an integer of the first four characters, so each season would be referenced by the year it started (2008 for 2008/2009 season). This would allow for easier and cleaner plotting by season.

In order to assess a team's performance on a yearly basis, I created a function that outputs an end-of-season league table to display what place each team finished in a given year and given league. These league tables also display each team's wins, losses, draws, goals scored, and goals conceded for the given year. I used this function to create another function that outputs where a given team placed in a given year.

The transfer data is contained in a collection of CSV files grouped by year and league. I looped through each year, read in the CSV of each league's transfers, appended each dataframe to a list, and then concatenated the list together to make a single dataframe containing all of the transfer data for the time period. This dataset includes the year, league, teams involved in the transfer, whether it is a purchase or sale, player name, player position, and the fee paid.

To make the transfer data more manageable, I first split the dataset into two subsets, whether the transfer was a purchase or sale. There were too many specific values for player position, so I replaced each value with Forward, Midfield, or Defender where appropriate. I also added an "age_range" column which had the value 15-22, 23-29, or 30+ depending on age. This would allow for future grouping to see how teams spent differently on different positions and age groups.

In order to match a team's match data to the corresponding transfer data, I added an ID column to the transfer data and populated it with the ID from the match dataset. This only worked, however, with team names that were exact matches across the two datasets, as some team names differed slightly from the different sources. For example, "Arsenal FC" is sometimes referred to simply as "Arsenal", which Python sees as different strings despite referencing the same team. In order to populate the ID column for teams without perfect matches, I used the partial_ratio function from the fuzzywuzzy.fuzz package to obtain the string comparison ratio and populated the ID number column if the strings met the necessary threshold.